



While much of the content is generic
could apply to any platform
I'm assuming a .NET environment, and where a language is necessary, C#

Focus on Code, not on other things; not that other things are irrelevant, but need to
focus somewhere.

Apologies in advance to Visual Basic Devs, though most stuff will still be relevant

Also not focussing on Governance stuff – selecting platform, vendor etc

License

This presentation licensed under the
Creative Commons Attribution License
<http://creativecommons.org/licenses/by/3.0/>



Towards Maintainability

What do I Need
When **the Maintainer** is me?



4 Key Terms

Though not exclusive, these are useful.

Perspicuous

Clarity
Don't Repeat Yourself

Discoverable

Make it easy to navigate

Principled

Every problem has multiple solutions
All software is opinion

Safe

No booby traps, sinkholes or mazes



Value, not Obstruction

Keep the Goal in mind:
Maintainable, reliable systems

This is not about red tape, or getting in the way.

It's all about value.

Choose processes that work for you, for your team, in your business.

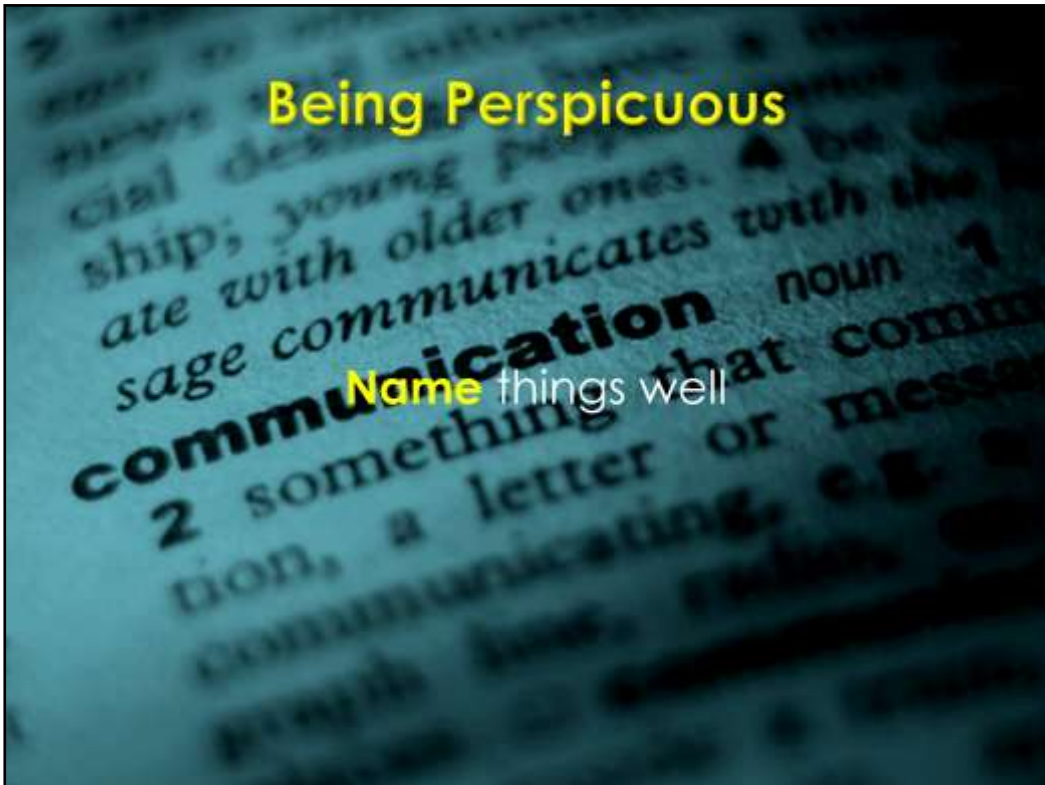
Change one thing at a time, get used to it.
Keep what works, eliminate what doesn't.



Clarity
Intelligibility

Useful to have Standards & Conventions

All about clear communication with other humans
Including people who aren't developers
System Administrators, DBAs, Help Desk, Trainers, and yes, Users too



Naming

of types, of methods, of members, of locals

Naming is important.

Clarity, Accuracy, Intelligibility and Reliability

Names are the first aspect encountered by other developers,
may be the most persistent and long lived aspect,
will be present even if everything else is lost

Names need to be clear and accurate (singular vs plural).

Trustworthy

Appropriate to Culture – “C” prefix in MFC

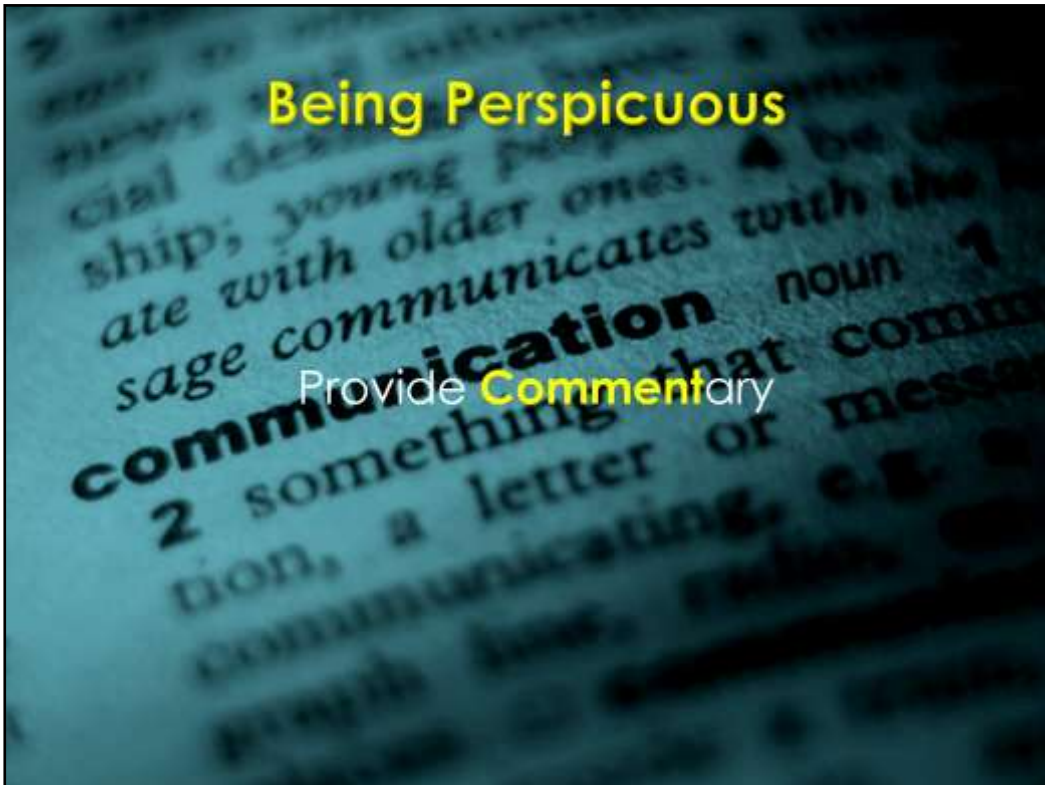
Ubiquitous Vocabulary

Design Patterns

www.definr.com

Being Perspicuous

Demos: **FileExists**, **GetSupplier**



We call talk about commenting our code, but often the comments are rubbish.

Anyone seen comments that were just plain wrong?

Comments should be intentional, not declarative.

Don't Repeat Yourself –
comments shouldn't indicate what is going on, but why we need to do it.

Don't just write things that can be worked out by reading the code
– give more information, things that otherwise would be guessed at.

Include references – to documents, websites – so that others can learn what you know.

If something might need to be improved, leave reminders to later self

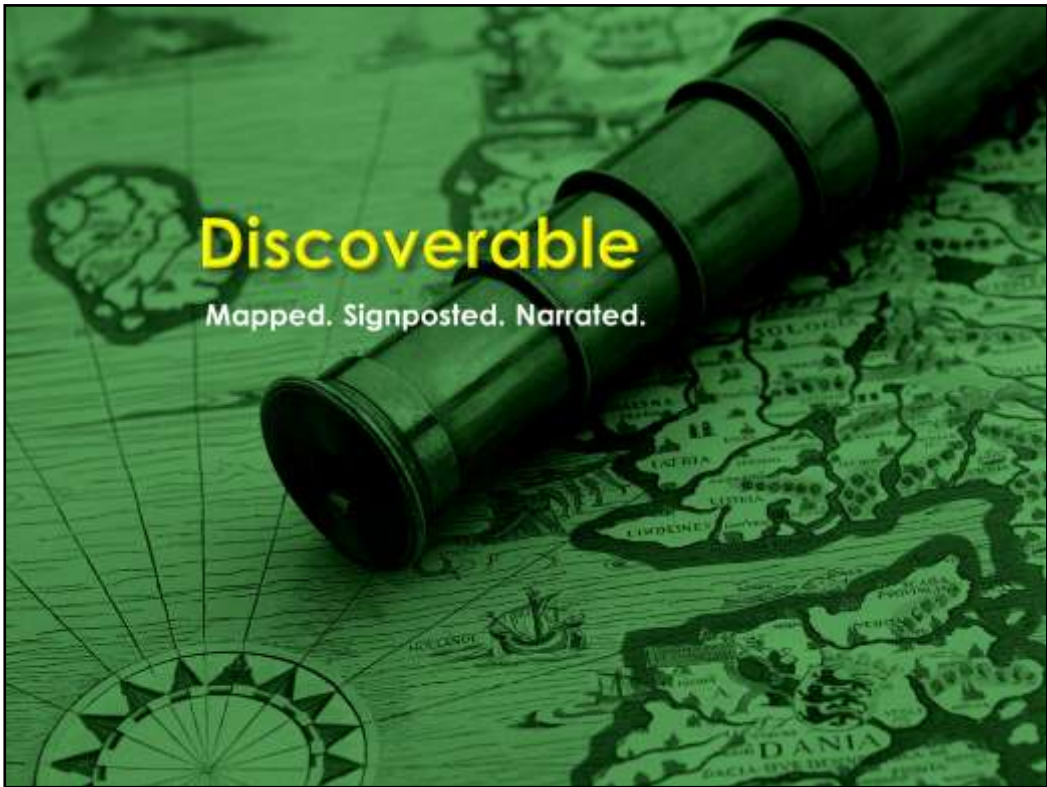
If something didn't work, leave notes so the next person doesn't repeat the mistake

Not everything needs commenting – good naming helps a lot.

Being Perspicuous

communication noun 1
2 something that commu-
nicates with the
sage communicates with the
ship; young people
cial desirability
news
ate with older ones. A be
nouns

Demo: Reader



Make it save to Explore

Don't let users get lost in your code



Be Tidy – and Predictable

Starting from a freshly paved machine,
what needs to be installed in order to work on your system?

Which tools are needed, which third party libraries, where is the source code to be found?

Is any of this written down – or is it part of the oral tradition.

Documentation of Architecture and Design

Tools to install

Standard folder structures

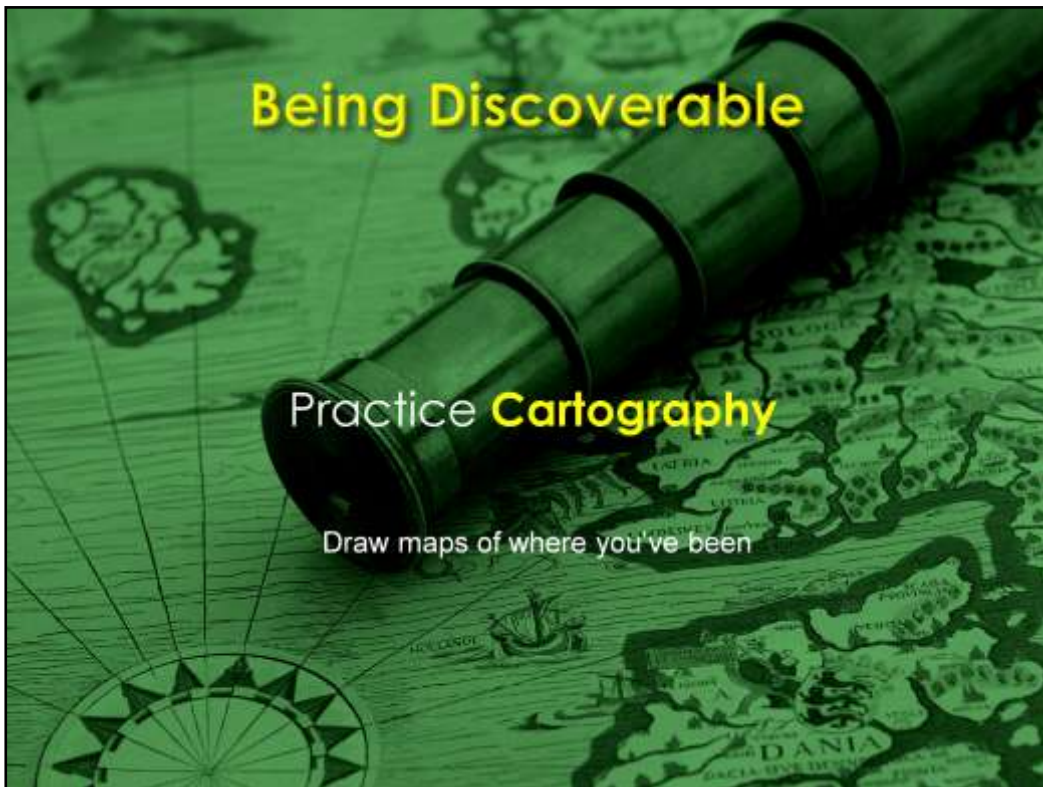
Libraries checked into source control

=

Being Discoverable

Demo: **Directory Structure**





Practice Cartography

== Creation of Maps

One of the best ways to find your way around a new city is with a map
class diagrams can be maps to your system.

Don't try to put everything on one diagram
use multiple diagrams, each detailing a single perspective.

Create Documentation

Architecture – map where everything belongs, layer by layer

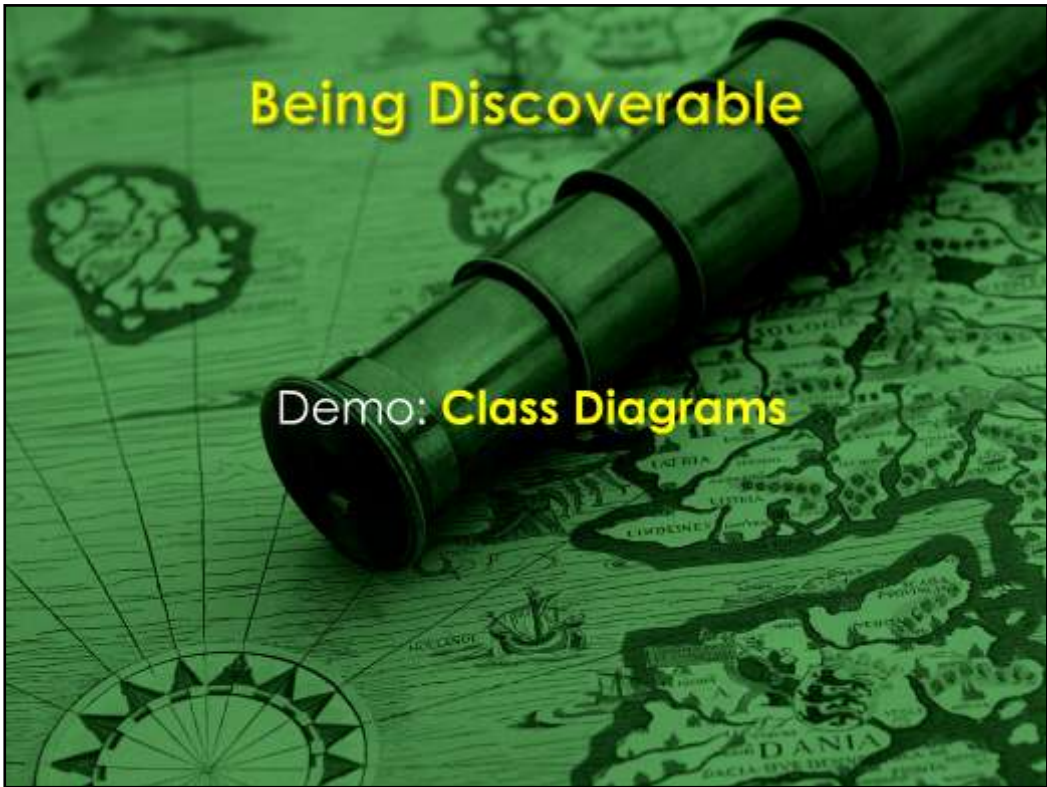
Functionality – requirements, specifications

User/System Docs – map of system structure, functional view

Support Docs – where everything is deployed – machine/folder

Being Discoverable

Demo: **Class Diagrams**





Code as Opinion

Know what you are doing, and why you are doing it

Always be willing to give a defence of what you believe.



Always do things well

Sometimes difficult to define "best" – religious arguments

"Well" changes slowly – not following the latest hype or fashion – a commitment to quality

Commit to doing things as well as you know how –
if you are going to cut corners, know which corners you're cutting and why.
Learn from other peoples examples, and always work on improving yourself.

Tension with maintaining consistency – need to balance between the two
Make changes, but on purpose, not by accident

Peer reviews useful



fxCop

Static Analysis for compiled assemblies.

Also available as Code Analysis feature in Visual Studio Team System editions.

Somewhat stagnant – infrequent updates.

<http://www.microsoft.com/downloads/thankyou.aspx?familyId=9aeaa970-f281-4fb0-aba1-d59d7ed09772&displayLang=en>



Gendarme

Open source alternative to fxCop, part of the Mono project.

Lively, though little known

<http://mono-project.com/Gendarme>

Recommend downloading the zip (cross-platform) version.



Snippets
tryf prog prog event



Write Opinionated Code

All code is opinion – you're stating that these steps are the ones to solve the problem

Don't be tentative about your opinions – know what you believe, and why.

Don't Work Too Hard

Learn the framework – take the time to learn –

Use the power of the framework – don't work harder than you need to.

Not only do you waste your own time, but you end up writing more code that someone else needs to understand.

Size of the framework – don't reinvent the wheel

Fail Fast

Your code should be opinionated –
if it needs A B & C to function properly,
it should throw a tantrum (exception) if it doesn't get it.

Discoverability

Major problem with development is finding things out

Talk with your peers

www.searchdotnet.com

Being Principled

Demo: **Xml Api**





Make it safe for someone to work on your code.

Don't let them get hurt.



Avoid Temporal Coupling

Demo.

Elements of Responsibility

Who is responsible for ensuring that things are performed in the right order

The object should take responsibility for itself, not be reliant on others

Single Responsibility Principle

Object Oriented vs Procedural Development

Code Smell: Object Envy



Being Safe

Demo: Updater



Keep things Consistent

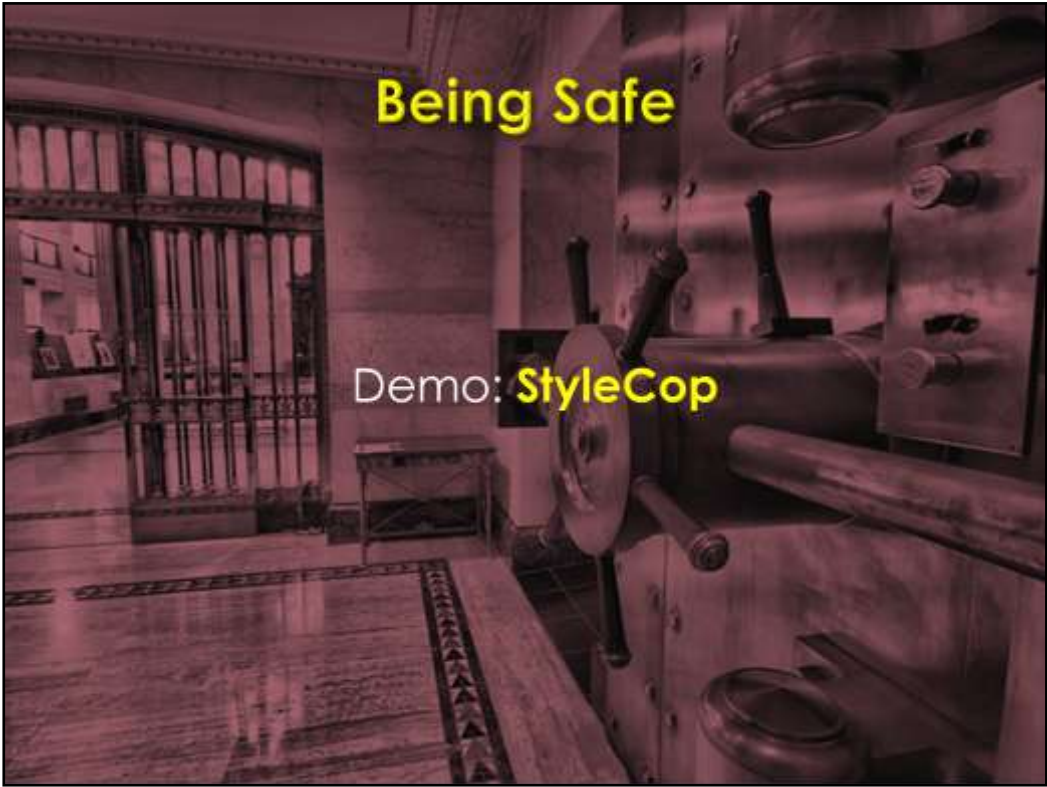
Consistency builds trust; be careful not to violate that trust

Compiler Warnings

- Use the compiler to your advantage
- Target zero warnings – use Warnings as Errors

Coding conventions

Doing things the best we know how



Being Safe

Demo: **StyleCop**





One thing at a time

Don't rush into making wholesale changes. Take things slowly, changing only one thing at a time. Keep what works, discard things that give you no value. Keep the goal in mind and work towards it – aim to be a better developer next month than you were last month.

Visual Studio

Pay attention to the tools you already have and learn to exploit them. Visual Studio is more than a glorified text editor.

Have the conversations

While you can change some by yourself, start the conversations with other people on your team. Work together to improve. But don't get sucked into the big bang approach ...

Automate stuff

Write command (batch) files; or use a tool. Get rid of the manual checklists and automate things as much as you can. Don't rely on a simple 45 point checklist for your builds. If it can be automated, it should be automated.

NAnt is a good tool, but so are MSBuild and Powershell. Can you build your entire system from a single command? If not, why not? Start with simple compilation, then build from there to include documentation, installers, etc etc.

Dictionary

Start paying more attention to your naming.

Writing

write some documentation. Add comments to your code, write up your design as a reference document, capture what's in your head and put it on paper.

Unit Testing

Start writing some unit tests



TestDriven.Net

Reduce the friction of unit testing by using TestDriven.Net.

FxCop

Learn from other peoples mistakes;
use FxCop to highlight potential problem areas.
Each warning should either be fixed or explicitly excluded –
never ignore a message. Goal is to have zero warnings.

Gendarme

An open source version of FxCop.
Checks different things, worth using as well.

StyleCop

Rigorous enforcement of C# coding style.
Lots of rules, some are wrong (IMHO).
Turn off the rules that cause you pain, and adhere to the rest.
Also plug in for Resharper.

Standcastle

Build documentation from your source code, in the style of MSDN.
Plain Sandcastle is hard to drive, pick one of the open source projects
that make it easier – such as SandCastle Help File Builder.

TeamCity

If you don't have a continuous integration server already, set up TeamCity.
The Professional edition is free, with minor restrictions.

Resharper

Productivity tool that integrates within Visual Studio.
Has to be experienced to be believed – makes lots of small things easier and simpler.



Towards Maintainability

Questions?



Bevan Arps

bevan@nichesoftware.co.nz
<http://www.nichesoftware.co.nz>